Bachelor thesis

Clustering of images using Generative Adversarial Networks

Faculty of Science and Engineering

Author:
Eloy MARÍN CIUDAD

Supervisors:
Emanuel SANCHEZ AIMAR, LiU
Amanda BERG, LiU
Philippe SALEMBIER, UPC

Examiner:
Jörgen AHLBERG, LiU

May 2020

Abstract

Recently, a variation of GANs, called ClusterGAN, has shown the ability to perform unsupervised classification of images according to content thanks to the use of a discrete-continuous latent space and a clustering-specific loss. This thesis demonstrates that it is possible to cluster images and annotate query images using GANs with some limitations according to image content. These limitations are related to the level of the features that describe an image, since clustering is performed based on low-level features. Also, in this thesis, we propose a new approach to cluster images by their dominant color, obtaining promising results. The proposed approach consists in using several small patches per image and the CIELAB color space because it approximates the way the human vision works.

Resum

Recentment, una variació de les GANs (ClusterGAN), ha demostrat l'habilitat de classificar imatges segons el seu contingut de forma no supervisada, gràcies a l'ús d'un espai latent discret-continu i una pèrdua especifica per dur a terme clustering. Aquesta tesi demostra que és possible agrupar imatges i anotar noves imatges utilitzant GANs amb algunes limitacions respecte al contingut d'aquestes. Aquestes limitacions estan relacionades amb el nivell de les característiques que descriuen una imatge, ja que el clustering es realitza basant-se en característiques de baix nivell. En aquesta tesi també proposem un nou enfocament per agrupar imatges segons el seu color dominant, obtenint resultats prometedors. L'enfocament proposat consisteix a utilitzar diversos pedaços per imatge i l'espai de color CIELAB, ja que aproxima la forma en la qual el sistema visual humà funciona.

Resumen

Recientemente, una variación de las GANs (ClusterGAN), ha demostrado la habilidad de clasificar imágenes según su contenido de forma no supervisada, gracias al uso de un espacio latente discreto-continuo y una perdida especifica para clustering. Esta tesis demuestra que es posible agrupar imágenes y anotar nuevas imágenes utilizando GANs con algunas limitaciones respecto al contenido de estas. Estas limitaciones están relacionadas con el nivel de las características que describen una imagen, ya que el clustering se realiza basado en características de bajo nivel. En esta tesis también proponemos un nuevo enfoque para agrupar imágenes según su color dominante, obteniendo resultados prometedores. El enfoque propuesto consiste en utilizar diversos parches por imagen y el espacio de color CIELAB, ya que aproxima la manera en la que el sistema visual humano funciona.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation

Over the recent years, the main focus of research in image clustering has been to transform the input data space to a latent space where the input features are disentangled, therefore the separation of data is easier, providing both dimensionality reduction and clustering.

GANs [1] have already shown the ability to learn meaningful disentangled representations in an unsupervised manner [2], providing a mapping from a latent space to the input data space. This latent space not only provides dimensionality reduction, but also gives rise to other applications such as clustering. The reason to use GANs is because changes in the latent space lead to changes in the semantic qualities of the generated images, giving GANs a remarkable interpretability.

Recently, [3] proposed ClusterGAN, a mechanism that shows the ability to cluster images in the latent space according to content, that is, to perform unsupervised classification of images using GANs. One possible application is to facilitate the annotation of unlabeled images. This is important since the classification performance of deep neural networks is commonly limited by the amount of manually annotated training data. In many cases, the process of annotating a dataset can be very time consuming, so it would be ideal if somehow the workload of this process could be reduced. Since GANs are a very powerful tool, another possible application is to perform complex clusterings, which leads us to one of the principal motivations of this work: *Can we propose to cluster images by their dominant color using GANs?*.

1.2 Aim

The main goals of the thesis are to evaluate the clustering quality using GANs on a varied range of datasets to find any limitations that can lead to poor clusterings, demonstrate if GANs can be used to annotate unlabeled images and propose a new approach to cluster images by their dominant color.

1.3 Research questions

This thesis aims to answer these two closely related research questions:

1. Given a large set of (manually classified) image samples belonging to several classes, can a GAN be used to sort the samples appropriately without having access to the class annotations? If so, can it cluster colors?

2. Once trained, can the GAN be used to label a query image by searching for the closest match in the latent space?

1.4 Delimitations

This thesis does not aim to develop a new framework or to improve [3] method to cluster images using GANs due to time limitations and limited resources.

1.5 Approach

The approach taken for answering the defined research questions consists in using ClusterGAN [3] implementation in Python as a baseline to train different models and gathering multiple datasets with different complexities to perform the desired experiments.

1.6 Thesis Outline

The planned procedure for this thesis is the following: Firstly, extensive research on the topics that this thesis works with will be performed in chapter 2. There, ClusterGAN [3] method will be described in detail and some approaches related to the task of clustering colors will be mentioned. Following, the datasets, hyperparameters, and architectures used will be described in chapter 3 as well as the chosen metrics to evaluate the clustering quality. Finally, the results of the experiments will be shown and analyzed in chapter 4. At the end, some conclusions and possible future work will be mentioned in chapter 6.

2. Background

This chapter describes the main theoretical concepts that this thesis works with, which are clustering, GANs [1] and clustering using GANs.

2.1 Clustering

Clustering is an unsupervised task used in many fields such as machine learning, image analysis and pattern recognition. It consists in grouping a set of data objects in an n-dimensional space, with the final objective of finding patterns in the dataset and clustering them in the same group. Ideally, data objects in the same group (called a **cluster**) would be more similar to each other than to those in other clusters.

There are a great variety of algorithms that solve this task with different understandings of what defines a cluster and how to find them. Some notions of clusters are groups with small distances between its members [4], dense areas of the data space [5] or statistical distributions [6].

The choice of distance measures is a key factor since it defines how the similarity between two data objects is calculated. Generally, the most used distance measure when clustering is the euclidean distance. Euclidean distance between two points P and Q in an n-dimensional space is defined as follows:

$$d(P,Q) = \sqrt{\sum_{i=0}^{n-1}(p_i - q_i)^2} \tag{2.1}$$

Also, the relation between clusters can be described in different ways. For example, *hard clustering* [7] assumes that a data object can only belong to one cluster while *soft clustering* (also: *fuzzy clustering*) [7] assumes that each data object belongs to each cluster to a certain degree.

The clustering algorithm and parameter settings (such as the distance function) to use depend on the data and the posterior use of the clustering result.

2.1.1 Clustering techniques

As mentioned before, the notion of a cluster varies depending on the cluster model used. A classical approach is centroid-based clustering.

In centroid-based clustering, clusters are represented by a central vector and the data objects are assigned to the clusters based on their proximity. This proximity is calculated, by default, using the euclidean distance (equation 2.1) between a data object and a cluster centroid.

K-means clustering [4] aims to group n data objects into k clusters in which each data object belongs to the cluster with the nearest centroid. One of the drawbacks of k-means clustering is that the number of clusters must be specified in advance.

The algorithm procedure consists of four main steps:

1. Initialize k random centroids
2. Assign each data object to the nearest centroid, creating clusters
3. Update the centroid of each cluster as the mean of the objects belonging to it
4. Repeat 2 and 3 until stopping criteria is met

Mathematically, k-means objective is to minimise its cost function, which is the sum of squared distances:

$$J = \sum_{j=0}^{k-1} \sum_{\substack{i=0 \\ x_i \in c_j}}^{n-1} ||x_i - \mu_j||^2 \tag{2.2}$$

x_i is a data object belonging to cluster j and μ_j is the centroid of cluster j.

Input: k (the number of clusters),
D (a set of lift ratios)
Output: a set of k clusters
Method:
Arbitrarily choose k objects from D as the initial cluster centers;
Repeat:
1. (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
2. Update the cluster means, i.e., calculate the mean value of the objects for each cluster

Until no change;

Figure 2.1: K-means pseudo-code. Source: Researchgate[1].

[1] https://www.researchgate.net/figure/The-pseudo-code-for-K-means-clustering-algorithm_fig2_273063437

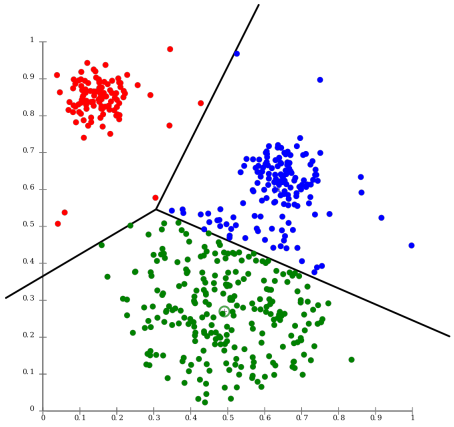


Figure 2.2: Example of k-means clustering. Source: Wikipedia[2].

2.2 Generative Adversarial Networks

A Generative Adversarial Network (GAN) [1] is a framework where two neural networks are simultaneously trained using an adversarial process. The generator G generates data samples that try to capture the real data distribution, while the discriminator D evaluates the probability of a sample coming from G (fake sample) or the real data (real sample).

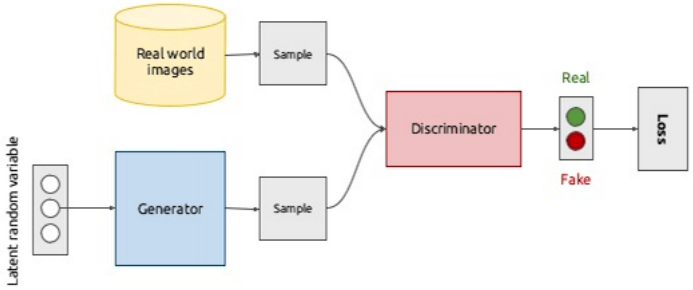


Figure 2.3: GAN architecture. Source:[3].

[2] https://commons.wikimedia.org/wiki/File:KMeans-Gaussian-data.svg

[3] https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

2.2.1 The generator

G generates samples by sampling a random vector from a latent space Z. This vector is used as an input to generate a data sample $x = G(z)$ that can "fool" D. The generator G goal is to mimic the training data distribution, therefore maximizing the probability of D making a mistake.

Usually, G is a Convolutional Neural Network (CNN) that projects z from the latent space onto the data space to generate a data sample x.

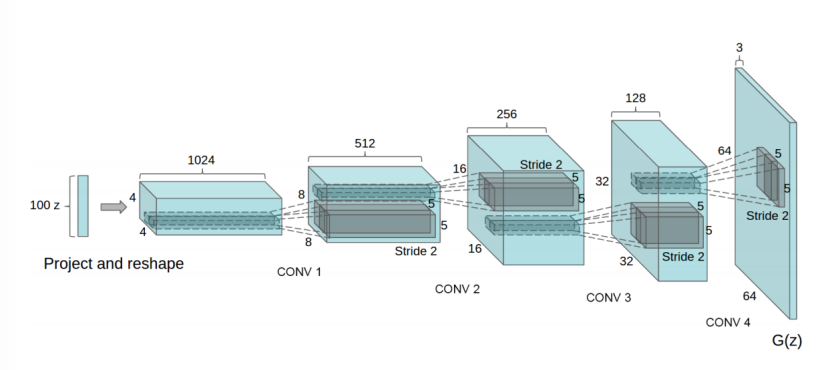


Figure 2.4: DCGAN Generator network. Source: Radford et al. [8].

G's objective is to minimize the probability of a generated sample being recognized as fake by D. This can be described as:

$$\min_G V(G) = E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.3}$$

2.2.2 The discriminator

D is trained to classify an input sample as real or fake, learning which features define a data sample as real. Then, by D learning these features, G will learn them too in order to achieve the objective of "fooling" D. Therefore, D can be seen as a guide for G to generate realistic samples. More specifically, D can be seen as a loss function of G's performance. Without D, G would just generate random samples.

Usually, D is a CNN that predicts the probability of a sample being real. The output is in range between 0 (predicted as fake) and 1 (predicted as real).

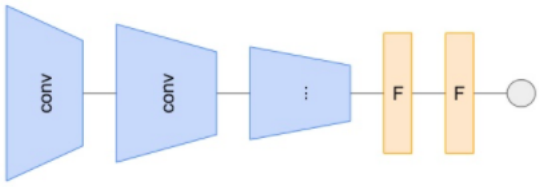


Figure 2.5: Discriminator network. Source:[4].

D's objective is to maximize the probability of recognizing real samples as real and fake samples as fake. This can be described as:

$$\max_D V(D) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.4}$$

2.2.3 Procedure

In sections 2.2.1 and 2.2.2, G and D objectives are described separately. However, they are jointly trained, so GANs can be described as a minimax game between G and D where the two networks compete to achieve their objective. This minimax game can be described as:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.5}$$

The training algorithm described in [1] that optimizes equation 2.5 consists in alternating between training D and G.

Firstly, G weights are fixed and we train D for k steps using both real and generated samples. In each step, we backprop the error to update D weights. Secondly, D weights are fixed and we use generated samples as D's input to finally backprop error to update G weights.

[4] `https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016`

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 2.6: GAN training algorithm. Source: Goodfellow et al. [1].

2.3 Clustering using GANs

Over recent years, with the use of deep learning techniques, the main focus of research in image clustering has been to transform the input data space to a latent space where the input features are disentangled, therefore the separation of data is easier.

Current clustering methods using deep learning techniques on images involve autoencoder-based methods [9, 10, 11, 12] and generative-based methods such as variatonal autoencoders and GANs.

Within the generative models family, Variational Autoencoders (VAE) [13] and Generative Adversarial Networks (GAN) are considered the most prominent way of solving the problem of joint clustering and dimensionality reduction.

InfoGAN's [2] objective is to learn disentangled representations by changing the latent space Z structure. Instead of only noise, a latent code is added as an input to G so it can have meaningful effects on the generated samples.

For example, for the MNIST handwritten digits dataset [14] that consists of 10 digits, the latent code would be a one-hot encoded vector of length 10. The idea is that keeping the same latent code and changing the noise input will produce different samples of the same digit.

To do so, [2] uses a GAN whose goal is to maximize the mutual information between the latent code and the generated sample.

While InfoGAN [2] learns disentangled representations, it is not designed for clustering. On the other hand, VAEs have the advantage of having an encoder, which enables mapping from the data space X to a lower-dimensionality space Z that could be clustering-friendly.

GANs have demonstrated the capacity to generate realistic samples but unfortunately, GANs do not have a network that allows for clustering.

To bridge the gap, a variation of GANs called ClusterGAN [3] has shown the ability to perform unsupervised clustering on images by joining an encoder and the concepts proposed by Xi Chen et al. [2].

2.3.1 ClusterGAN

ClusterGAN [3] is proposed as a new mechanism for clustering using GANs. To do so, they propose three main algorithmic ideas:

- Use of a mixture of discrete and continuous latent variables, inspired on [2].
- Use of an explicit inverse-mapping network (encoder $\mathcal{E}$) to obtain the latent variables given a data sample.
- Joint training of the GAN and the encoder with a clustering-specific loss.

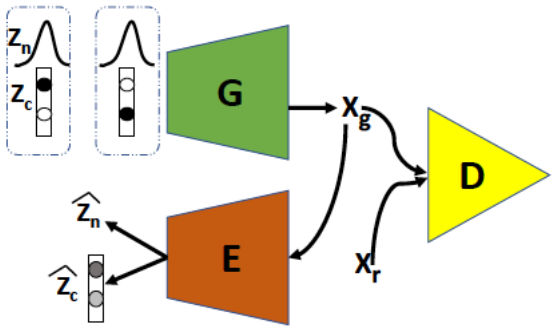


Figure 2.7: ClusterGAN architecture. Source: Latent Space Clustering in Generative Adversarial Networks [3].

The encoder network $\mathcal{E}$ that does the inverse process of G, mapping the data space to the latent space $X \to Z$, has also the same architecture than G but inverted.

To ensure a precise reconstruction of the latent vectors, the clustering specific-loss term is added to the minimax objective of GANs:

$$\begin{aligned}\min_{G,\mathcal{E}} \max_{D} V(D,G,\mathcal{E}) =& E_{x\sim p_{data}(x)}[q(D(x))] + E_{z\sim p_z(z)}[q(1-D(G(z)))] \\ &+ \beta_n E_{z\sim p_z(z)}[||z_n - \mathcal{E}(G(z_n))||^2] + \beta_c E_{z\sim p_z(z)}[\mathcal{H}(z_c, \mathcal{E}(G(z_c))]\end{aligned} \tag{2.6}$$

where $\mathcal{H}$ is the cross-entropy loss and $q(.)$ is a quality function. In Vanilla GAN, $q(x) = log(x)$ and in Wasserstein GAN (WGAN) [15] $q(x) = x$. β_n and β_c are coefficients that enable to choose the importance given to the precise reconstruction of continuous and discrete variables of the latent space.

This latent space can be formally described as $z = (z_n, z_c)$, $z_n \sim \mathcal{N}(0, I_n)$ and $z_c = e_k, k \sim \mathcal{U}[1, ..., K]$ where e_k is the k^{th} one-hot encoded vector in $\mathbb{R}^K$ and K is the number of clusters. Ideally, each e_k will generate samples corresponding to a single real class.

Mukherjee et al. [3] obtained good clustering performance on several datasets such as MNIST [14] or Fashion-MNIST [16]. An interesting research question may be whether ClusterGAN can cluster images by their dominant color.

2.3.2 Clustering colors

Van de Weijer et al. [17] approached this issue proposing to learn color names from real-world images with a weakly supervised approach and outperforming other methods previously used such as chip-based color naming.

To do so, they used a generative model called Probabilistic Latent Semantic Analysis (PLSA) [18] and used the L*a*b* [19] color space because it approximates how human vision works, it is perceptually linear. L*a*b* is designed so that the same amount of numerical change in these values correspond to the same amount of visually perceived change.

Van de Weijer et al. [17] use the 11 basic color terms of the English language: black, blue, brown, grey, green, orange, pink, purple, red, white and yellow.

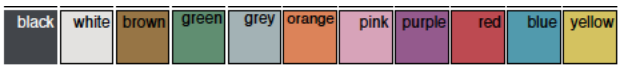


Figure 2.8: Prototypes of the 11 color terms learned. Source: van de Weijer et al. [17].

Given the progress that has taken place in this research field, in this work we propose to evaluate the performance of [3] for the task of colors clustering.

3. Method

In this chapter, the method used in the conducted experiments is presented. Firstly, the datasets used will be presented. Next, the architecture and hyperparameters chosen for each dataset will be described and finally the metrics used to evaluate the clustering quality will be explained.

3.1 Datasets

This section describes the different data sets that this thesis works with. We used datasets of different complexities to evaluate how well images can be clustered when their features are of increasing complexity. In other words, we wanted to evaluate any limitations when clustering due to the complexity of the datasets. These datasets are the following:

- **MNIST handwritten digits**: The MNIST dataset [14] contains 28x28 grayscale images of handwritten digits between 0 and 9. This dataset is often used as a benchmark to validate machine learning algorithms.

Figure 3.1: Sample images from MNIST. Source:[1].

- **Fashion-MNIST**: Fashion-MNIST [16] is a dataset designed to replace the original MNIST dataset [14], increasing its complexity, for benchmarking machine learning algorithms. The dataset contains 28x28 grayscale images that can belong to 10 different classes: *t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot.*

[1] `https://en.wikipedia.org/wiki/MNIST_database#/media/File:MnistExamples.png`

Figure 3.2: Sample images from Fashion-MNIST. Source:[2].

In this work we will use two versions of this dataset: The original Fashion-MNIST dataset (which we will refer to as **Fashion-10**) and a modified version that groups images into 5 classes as **Fashion-5**. Fashion-5 is proposed in [3], merging some categories which are similar to form a 5-class dataset. These 5 classes are grouped as: *[t-shirt/top, dress], [trouser], [pullover, coat, shirt], [bag], [sandal, sneaker, ankle boot].*

- **CIFAR-10**: CIFAR-10 dataset [20] consists of 32x32 color images in 10 classes: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.*

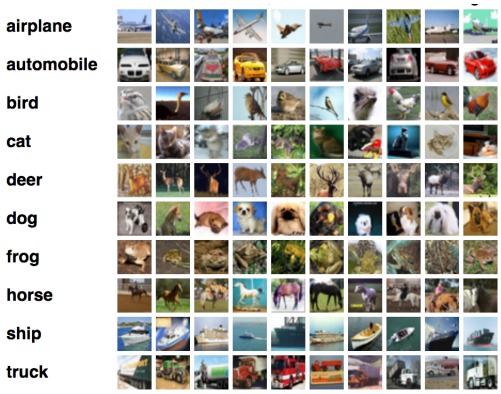


Figure 3.3: Sample images from CIFAR-10. Source:[3].

- **Termisk**: This dataset consists of thermal images provided by Termisk Systemteknik [4]. The dataset contains 96x96 grayscale images in 15 classes.

[2] `https://www.kaggle.com/c/insar-fashion-mnist-challenge`
[3] `https://www.cs.toronto.edu/~kriz/cifar.html`
[4] http://termisksystemteknik.se/

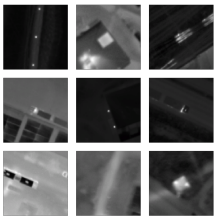


Figure 3.4: Sample images from Termisk dataset.

Data augmentation has been applied to the dataset, rotating each image several times, since they were captured from above (they are rotation invariant).

- **Color datasets**: For clustering images by their dominant color from real-world images, a test set of labeled images and a training set are required. Two data sets are used with that purpose, one of them collected specifically for this work. In section 3.1.1 these datasets are described in depth.

3.1.1 Color Data Sets

This section describes the data used to train and evaluate the performance of the model on the task of clustering colors:

- **Google images color name set:** The set was collected by using Google Images search engine. Also, to make the process of downloading images more efficient, Javascript was used to scrape images URLs and fastai Python library was used to download these URLs to their corresponding label directory. We use the 11 basic color names in the English language (see Fig. 2.8).

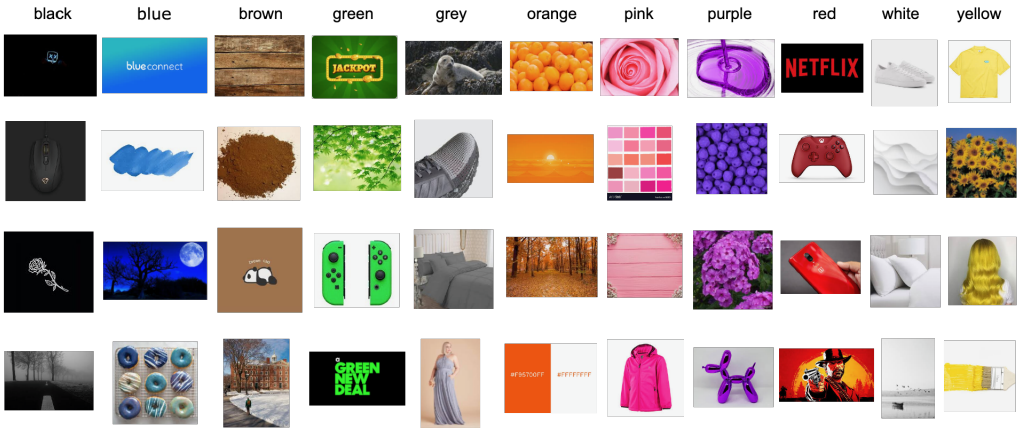


Figure 3.5: Google-retrieved examples for color names. For each color, 380 images were collected.

Once all the images are collected, the data that will be used as input to the model needs to be defined. We do not use raw images because the model expects images of a fixed size. Then, during the experiments we use different versions of the dataset as input data, changing two main factors: **color space** and the use of **image patches** or **resized images**. Used color spaces are RGB and CIELAB [19] because it is perceptually linear, approximating how human vision works. The reason for using image patches is that there is no need to resize the images, which can change the original color distribution.

- **Ebay color name set:** To test the clustering of colors, a human-labeled dataset is required. [17] used images from eBay, where users labeled their objects with a text description that often includes color information. The set contains four categories of objects (cars, dresses, shoes, and pottery) with 10 images per color and category. [17] hand-segmented the areas that corresponded to the color name for each image. Then, for each image, we extract five 32x32 patches from the mask area.

Figure 3.6: Ebay dataset example images. Only one mask example is shown per category. Source: van de Weijer et al. [17].

3.2 Procedure

To train a model, we used the implementation of ClusterGAN [3] authors[5].

3.2.1 Hyperparameter and Architecture details

As in [3], the networks are trained with the Adam Optimizer [21] (learning rate $\eta = 1e{-}04$, $\beta_1 = 0.5$, $\beta_2 = 0.9$) for all datasets and D is updated 5 times for each G update in training. The dimension of z_c is the same as the number of classes in the dataset, with some exceptions where we define more or fewer clusters than real classes. Details for each dataset are provided below:

- **MNIST and Fashion-MNIST**

  We used the same hyperparameters and architecture as [3], batch size = 64, z_n of 30 dimensions for MNIST and 40 dimensions for Fashion-MNIST. LReLU activation with leak = 0.2 was used. $\beta_n = 10$ for MNIST and $\beta_n = 0$ for Fashion-MNIST, $\beta_c = 10$ for both.

Generator	**Encoder**	**Discriminator**
Input $z = (z_n, z_c)\ \epsilon\ \mathbb{R}^{d_{z_n}+d_{z_c}}$	Input $X\ \epsilon\ \mathbb{R}^{28x28}$	Input $X\ \epsilon\ \mathbb{R}^{28x28}$
FC 1024 ReLU BN	4 x 4 conv, 64 stride 2 LReLU	4 x 4 conv, 64 stride 2 LReLU
FC 7 x 7 x 128 ReLU BN	4 x 4 conv, 128 stride 2 LReLU	4 x 4 conv, 128 stride 2 LReLU
4 x 4 upconv, 64 stride 2 ReLU BN	FC 1024 LReLU	FC 1024 LReLU
4 x 4 upconv, 1 stride 2, Sigmoid	FC $d_{z_n} + d_{z_c}$ linear for $\hat{z}$ Softmax on last 10 to obtain $\hat{z_c}$	FC 1 linear

Table 3.1: Architecture for MNIST and Fashion-MNIST datasets.

- **Termisk**

  We used batch size = 64, z_n of 50 dimensions. LReLU activation with leak = 0.2 was used. $\beta_n = 10$ and $\beta_c = 10$.

[5] `https://github.com/sudiptodip15/ClusterGAN`

Generator	Encoder	Discriminator
Input $z = (z_n, z_c)\ \epsilon\ \mathbb{R}^{d_{z_n}+d_{z_c}}$	Input $X\ \epsilon\ \mathbb{R}^{64x64}$	Input $X\ \epsilon\ \mathbb{R}^{64x64}$
FC 2048 ReLU BN	4 x 4 conv, 32 stride 2 LReLU	4 x 4 conv, 32 stride 2 LReLU
FC 8 x 8 x 128 ReLU BN	4 x 4 conv, 64 stride 2 LReLU	4 x 4 conv, 64 stride 2 LReLU
4 x 4 upconv, 64 stride 2 ReLU BN	4 x 4 conv, 128 stride 2 LReLU	4 x 4 conv, 128 stride 2 LReLU
4 x 4 upconv, 32 stride 2 ReLU BN	FC 2048 LReLU	FC 2048 LReLU
4 x 4 upconv, 1 stride 2, Sigmoid	FC $d_{z_n} + d_{z_c}$ linear for $\hat{z}$ Softmax on last d_{z_c} to obtain $\hat{z_c}$	FC 1 linear

Table 3.2: Architecture for Termisk dataset.

- **CIFAR-10 and Colors**

  We used batch size = 64, z_n of 50 dimensions for CIFAR and 20 dimensions for Colors dataset. LReLU activation with leak = 0.2 was used. $\beta_n = 10$ and $\beta_c = 10$.

Generator	Encoder	Discriminator
Input $z = (z_n, z_c)\ \epsilon\ \mathbb{R}^{d_{z_n}+d_{z_c}}$	Input $X\ \epsilon\ \mathbb{R}^{32x32x3}$	Input $X\ \epsilon\ \mathbb{R}^{32x32x3}$
FC 2 x 2 x 448 ReLU BN	4 x 4 conv, 64 stride 2 LReLU	4 x 4 conv, 64 stride 2 LReLU
4 x 4 upconv, 256 stride 2 ReLU BN	4 x 4 conv, 128 stride 2 LReLU BN	4 x 4 conv, 128 stride 2 LReLU BN
4 x 4 upconv, 128 stride 2 ReLU BN	4 x 4 conv, 256 stride 2 LReLU BN	4 x 4 conv, 256 stride 2 LReLU BN
4 x 4 upconv, 64 stride 2 ReLU BN	4 x 4 conv, 512 stride 2 LReLU BN	4 x 4 conv, 512 stride 2 LReLU BN
4 x 4 upconv, 3 stride 2, Sigmoid	FC $d_{z_n} + d_{z_c}$ linear for $\hat{z}$ Softmax on last d_{z_c} to obtain $\hat{z_c}$	FC 1 linear

Table 3.3: Architecture for CIFAR-10 and Colors dataset.

3.3 Metrics

In order to measure the performance of a clustering, we use as cluster indexes the maximum argument of $\hat{z_c}$. That is because G maps each z_c to a certain set of samples or mode[6], already deciding which

[6] A mode is the set of samples generated by the same z_c

type of images are clustered together. Then, $\mathcal{E}$ reconstructs the latent space $\hat{Z}$, where images from the same mode should be grouped into the same $\hat{z}_c$. Formally, the cluster assignment can be described as:

$$w_k = \arg\max \hat{z}_c \tag{3.1}$$

where w_k belongs to the set of clusters W.

So, to be able to properly cluster images, G must generate modes that may or may not correspond to real labels, as long as they have meaningful differences between them. Then, $\mathcal{E}$ must properly reconstruct the latent space in order to maintain this meaningful difference between modes.

Furthermore, the evaluation of the quality of a clustering is as important as the clustering task itself. Different approaches exist to try to solve the problem of evaluating a clustering, such as internal evaluation and external evaluation [22].

Given that we have "ground truth" labels for test data, we used an external evaluation approach, comparing the clustering to the ground truth labels.

We used the following external evaluation measures:

- **Purity:** Purity measures if clusters contain a single class. To be calculated, each cluster is assigned to the class that is most frequent in the cluster, then we take the sum over all clusters and divide by the number of data points N. Formally:

  $$purity(W, C) = \frac{1}{N} \sum_{i=1}^{k} \max_j |w_k \cap c_j| \tag{3.2}$$

  where w_k belongs to the set of clusters W and c_j belongs to the set of classes C.

  Good clusterings have a purity close to 1. One drawback is that high purity is easy to achieve with a high number of clusters. For example, putting each data point in its own cluster a purity of 1 is achieved. Therefore, purity cannot be used to compare clustering quality against the number of clusters.

- **NMI:** Normalized mutual information measures how much information is shared between a clustering and a ground-truth classification. Because it is a normalized measure, it enables us to compare the quality of clustering between clusterings that have different number of clusters. Formally, it can be described as:

  $$NMI(W, C) = \frac{2 * I(W; C)}{H(W) + H(C)} \tag{3.3}$$

  where $H(W)$ and $H(C)$ are the entropy of cluster labels and class labels respectively. I is the mutual information:

  $$I(W; C) = H(C) - H(C|W) \tag{3.4}$$

  NMI is a number between 0 and 1, where values close to 1 indicate a good clustering.

- **ARI:** Adjusted Rand Index measures how similar a clustering to the ground-truth classification is. Random clusterings are penalized, having an ARI close to 0 while 1 stands for perfect match. Formally, ARI can be described as:

$$ARI = \frac{\sum_{i,j}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}} \tag{3.5}$$

where i refers to the row number, j refers to the column number, a refers to the row sum and b refers to the column sum of the contingency table between the clustering and ground-truth labels.

	C_1	C_2	...	C_s	Sum
$W1$	n_{11}	n_{12}	...	n_{1s}	a_1
$W2$	n_{21}	n_{22}	...	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
W_k	n_{k1}	n_{k2}	...	n_{ks}	a_k
Sum	b_1	b_2	...	b_s	

Table 3.4: Contingency table.

- **Co-occurrence matrix:** A co-occurrence matrix can be used to quickly visualize the clustering assignment.

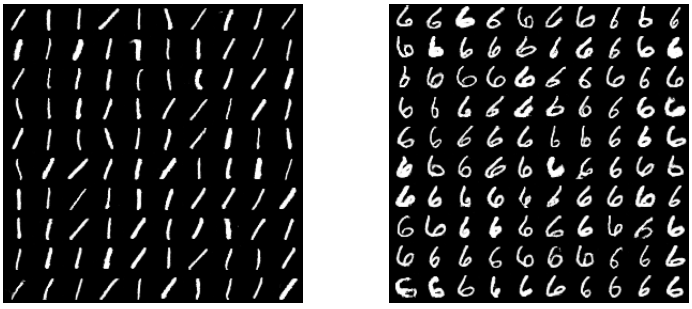
4. Results

4.1 Clustering capability

In order to investigate the clustering capability and limitations, we used different datasets with different complexities.

- **MNIST**

  For the MNIST [14] dataset, we trained a model with the architecture and hyperparameters described in section 3.2.1. Using this dataset, G learned to generate modes that corresponded to real labels (see Fig 4.1) and at the same time $\mathcal{E}$ learned to reconstruct G's latent space, providing an excellent clustering on the test set with $Purity = 0.938$, $NMI = 0.867$ and $ARI = 0.868$.



(a) mode 0 (b) mode 7

Figure 4.1: Generated digits from distinct modes (see Fig 6.1 for examples of all modes).

Then, we can visualize how the images were spread along the clusters using a co-occurrence matrix in Fig 4.2. We can see that $\mathcal{E}$ perfectly reconstructed the latent space because each digit has been assigned to a cluster that corresponds to G modes. Also, this matrix gives some interesting information, such as which two digits are the most similar for the trained model, in this case, 2 and 3.

Thanks to this matrix, we can also answer the second research question in section 1.3. It is possible to annotate unlabelled images since we know the mapping between clusters and real

labels. To annotate a query image, we use it as $\mathcal{E}$'s input, get the assigned cluster using eq. 3.1 and map it to its corresponding real label.

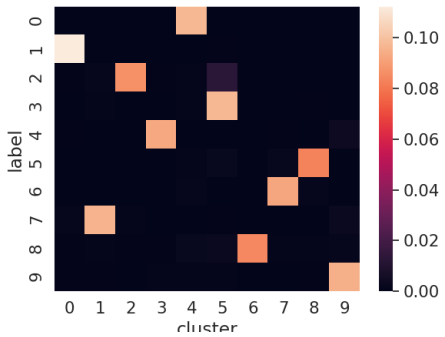


Figure 4.2: Co-occurrence matrix for MNIST dataset. Rows represent class labels and columns represent clusters.

Since MNIST is considered a simple dataset, now the challenge was to evaluate which types of images can be clustered based on their complexity.

- **Fashion-MNIST**

  For Fashion-MNIST [16], G learned to generate realistic images but some of the generated modes did not correspond to real labels. In fig 4.3, we can see a couple of examples where images that belong to different labels are grouped together. For instance, mode 1 groups sandal and bag images together.

(a) mode 1 (b) mode 9

Figure 4.3: Generated images from distinct modes. Shown modes contain images that have different labels.

As expected, the clustering performance is poorer than that achieved using the MNIST dataset, obtaining the following measures: $Purity = 0.576$, $NMI = 0.591$ and $ARI = 0.42$. Looking

at the co-occurrence matrix (see Fig 4.4), we can see that the clustering quality is not good because similar items, like boots and sneakers or coats and pullovers, are grouped together. That can mean that the model is able to differentiate between fashion items that belong to different "superclasses". For instance, sandals, sneakers and ankle boots can be grouped into a "superclass" that represents footwear.

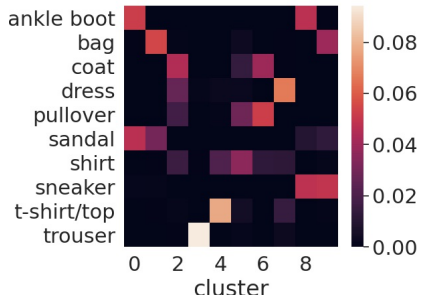
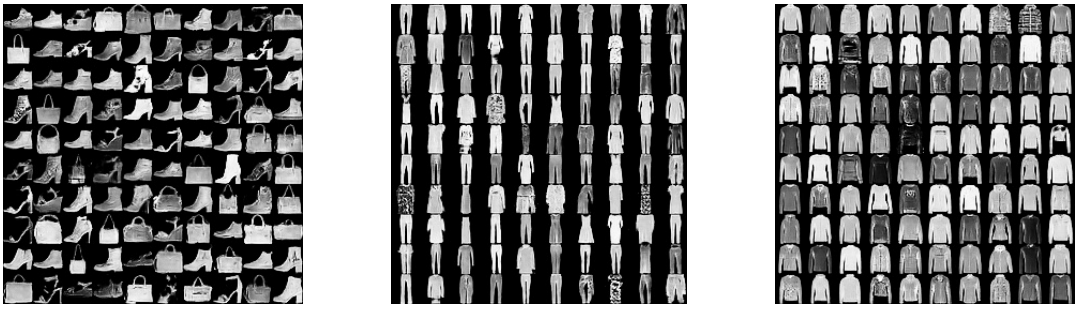


Figure 4.4: Fashion-10 co-occurrence matrix.

To evaluate if the model is able to differentiate between these "superclasses", we used the dataset version with 5 classes: (*[t-shirt/top, dress], [trouser], [pullover, coat, shirt], [bag], [sandal, sneaker, ankle boot]*), trying to enforce G to focus on learning features that differentiate dissimilar classes such as sandals/boots and bags. But, as we can see in Fig 4.5, that was not the case because footwear and bags were still grouped together and trousers label, that when using 10 clusters was assigned to a unique cluster, was now grouped with other classes.

The obtained metrics are the following: $Purity = 0.714$, $NMI = 0.57$ and $ARI = 0.45$. If we compare the obtained metrics between Fashion-10 and Fashion-5, we can state that both clusterings have a similar quality because the mutual information and the similarity between clusters and ground-truth labels is very similar for both cases.

These results already show that there are some limitations when clustering using GANs due to the complexity level of image features.



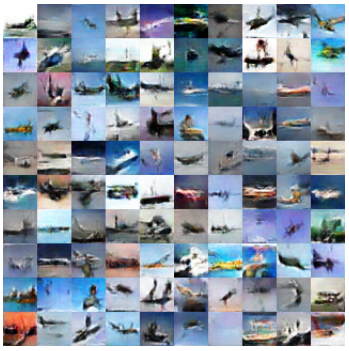
(a) mode 1 (b) mode 3 (c) mode 4

Figure 4.5: Fashion-5 generated images from distinct modes.

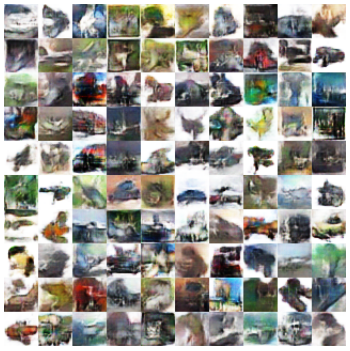
- **CIFAR-10**

  Clustering quality in the CIFAR-10 [20] dataset, a more complex dataset than MNIST and Fashion-MNIST, is really poor, obtaining $Purity = 0.21$, $NMI = 0.085$ and $ARI = 0.041$. If we compare it with a random guessing, that obtained $Purity = 0.11$, $NMI = 0.001$ and $ARI = 0$, we can see that both performances are very similar, so we can consider the clustering performed on the CIFAR-10 dataset as a random clustering.

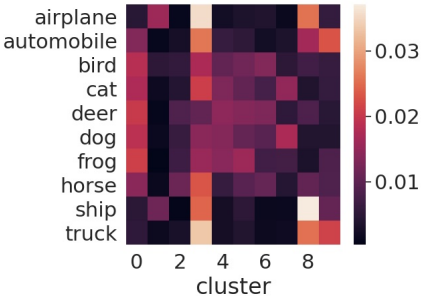
  The reason why the clustering performance is so poor is that G learned to generate modes based on low-level features, such as background color. G needed to learn high-level features because there is great intra-class variability in CIFAR-10 dataset. One example can be mode 1, where images of airplanes and ships are grouped together because the background color is blue (see Fig. 4.6a) or mode 3, where most images have a white background (see Fig. 4.6b). Another way to demonstrate that the model has not learned to differentiate real classes is by using the test set, visualizing the co-occurrence matrix (see Fig. 4.6c), where each cluster contains images from multiple classes.



(a) mode 1



(b) mode 3

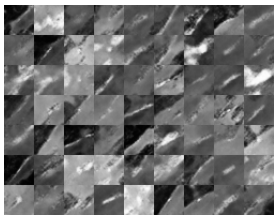


(c) Co-occurrence matrix

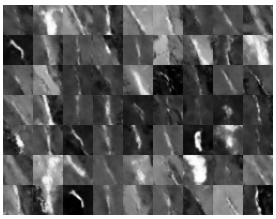
Figure 4.6: Generated images from mode 1 and co-occurrence matrix, showing that the model has not learned high-level features to properly cluster the images.

- **Termisk**

  As with CIFAR-10, when clustering the Termisk dataset, G learns to group images mainly based on their orientation (low-level feature), which makes sense given that each image in the dataset is rotated several times.



(a) mode 1



(b) mode 2

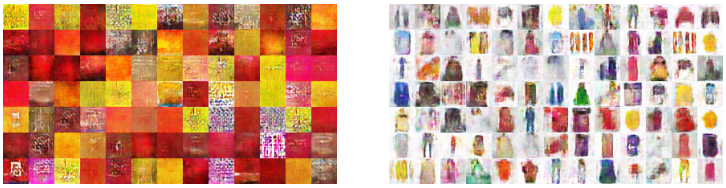
Figure 4.7: Generated images from distinct modes.

After these experiments, it was clear that ClusterGAN is able to cluster images but with some limitations. These limitations are related to unsupervised nature of clustering and the complexity of the images to cluster, understanding by complexity the level of the features that describe an image. Trained models learn to divide images based on low-level features because they are not forced to divide images based on high-level features, producing poor clusterings on complex datasets. For instance, with MNIST the clustering quality is excellent, but when more complexity is added, the model keeps clustering based on low-level features, such as background color for CIFAR-10 or orientation for Termisk.

Then, we proposed an approach for the task of clustering images by their dominant color, since it is a challenge that involves learning low-level features.

4.2 Clustering colors

Firstly, for the task of clustering colors, we resized images to a fixed size and used them as input for the model. The result was the following: G did not learn to properly differentiate between colors because it took into account other features such as white background or plain images (see Fig. 4.8b). Nevertheless, the results were promising, since it learned to group nearby colors in the RGB space like red, yellow, and orange (see Fig. 4.8a).

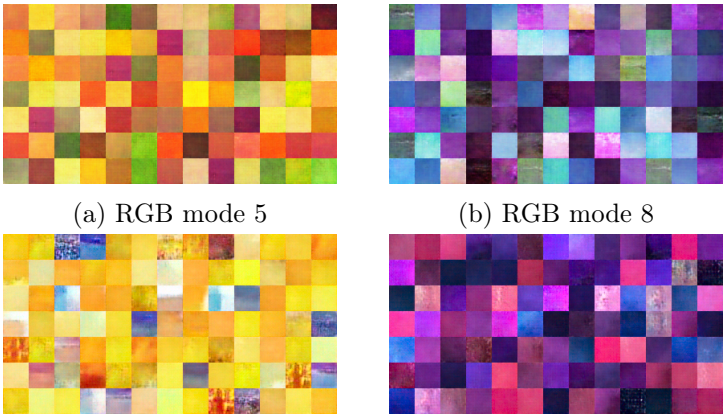


(a) mode 5 (b) mode 7

Figure 4.8: Generated modes using RGB resized images.

In order to enforce G to not differentiate images by other characteristics than their color, we pick a small patch (32x32) from each training image and use it as input for the model. We use small patches because we can assume that patches will only contain one color and be plain while giving more information about the image than a pixel-based approach.

When training the model using RGB patches, G performs better than when using resized images. Then, we decided to also use CIELAB color space because it approximates the way human vision works, CIELAB is designed so that the same amount of color change corresponds to the same amount of visually perceived change. As we can see in Fig. 4.9, using patches helps G with the task of grouping colors.



(a) RGB mode 5 (b) RGB mode 8

(c) CIELAB mode 0 (d) CIELAB mode 2

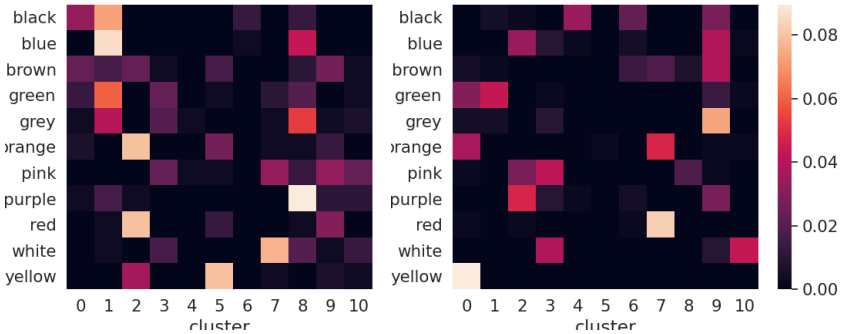
Figure 4.9: Generated images from distinct modes using: (a,b) RGB patches, (c,d) CIELAB patches. See Fig. 6.7 and Fig. 6.8 for examples of all RGB and CIELAB modes.

Then, to test the quality of clustering, we pick five 32x32 patches from the binary mask of each image in the test set (eBay dataset described in section 3.1.1) and use them as $\mathcal{E}$'s input. The cluster assigned to each image is the most frequent cluster assigned to the patches belonging to the same image. Both RGB and CIELAB patches have a poor clustering quality (see Table 4.1).

	Purity	NMI	ARI
RGB patches	0.27	0.21	0.07
CIELAB patches	0.32	0.25	0.106

Table 4.1: Clustering performance using 11 clusters.

If we take a look at their co-occurrence matrices (see Fig. 4.10), we can see that both models learned to group together close colors in their respective color space, but were not able to split them up into separate clusters. For instance, in Fig. 4.10b, red and orange are grouped together in cluster 7 and pink, purple and blue in cluster 2.

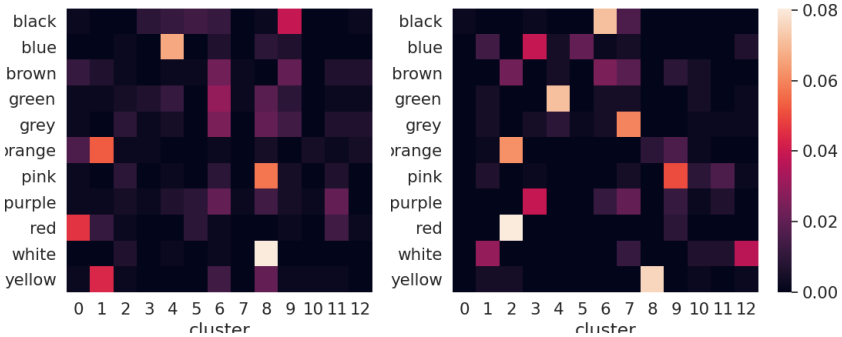


(a) Co-occurrence matrix using RGB patches

(b) Co-occurrence matrix using CIELAB patches

Figure 4.10: Co-occurrence matrices using patches and 11 clusters.

In order to enforce G to group similar colors into different clusters, we use a number of clusters k bigger than the 11 basic colors, since the number of basic color terms is ambiguous, specially between different languages [17]. With $k = 13$, we can see that some colors, such as pink in Fig. 4.11b or blue in Fig. 4.11a, are now grouped in a cluster by themselves. Although the majority of colors do not belong to a unique cluster, we can see that the clustering quality is better when using 13 clusters, obtaining a nice matching for CIELAB patches.



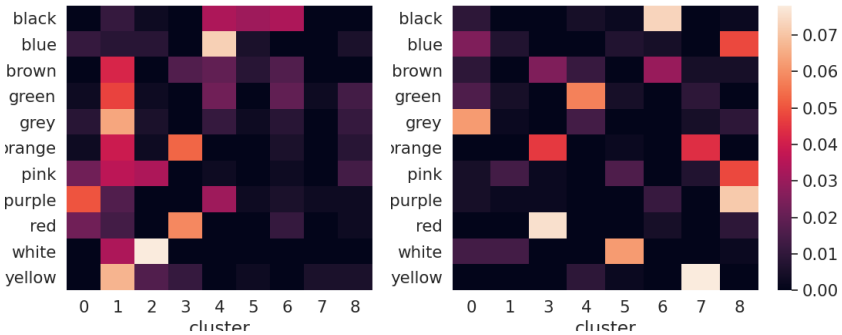
(a) Co-occurrence matrix using RGB patches

(b) Co-occurrence matrix using CIELAB patches

Figure 4.11: Co-occurrence matrices using patches and 13 clusters.

When using less clusters, $k = 9$, we can see in Fig. 4.12b that, in the case of CIELAB patches, we forced the model to group similar colors together, obtaining a nice match for these similar colors. For instance, some of the clusters contain the following colors: *cluster 8: [blue, pink, purple], cluster*

7: [orange, yellow], cluster 6: [black, brown]



(a) Co-occurrence matrix using RGB patches

(b) Co-occurrence matrix using CIELAB patches

Figure 4.12: Co-occurrence matrices using patches and 9 clusters.

Although the results obtained for the clustering of images by their dominant color show that the model cannot perfectly cluster colors, these results are promising since nearby colors are clustered together, so future work could lead to an improvement of the proposed approach. For instance, the performance of the model may have been affected by a noisy patch selection or other combinations of hyperparameters and architectures may lead to a better performance.

4.3 Annotation of query images

In order to evaluate if a trained model can be used to label a query image by searching for the closest match, we define two approaches. The first one consists in using the clustering assignment of an image as an annotation, and then mapping this assignment to the original label. The second approach consists in using K-Nearest Neighbours (KNN) [23] to find the closest match in the training set for a test sample. We can use as labels for the training set their ground-truth labels (supervised approach) or their cluster assignments.

To test these approaches we use the MNIST dataset because the clustering performance on this dataset was excellent. The mapping between cluster indices and original labels can be found in Fig. 4.2. The accuracy for each approach can be found in the following table:

	Cluster index	KNN using ground-truth labels	KNN using cluster indices
Accuracy	0.938	0.9615	0.938

Table 4.2: Accuracy on annotating test images using different approaches.

As we can see, it is possible to annotate query images once a model is trained, obtaining a good performance for the MNIST dataset. As expected, the approach with the best performance is the one using KNN with ground-truth labels, since similar images will be neighbours although they may be not assigned to the correct latent code $\hat{z}_c$. Also, using cluster indices and using KNN with cluster indices as ground-truth labels produce the same result because of the latent space discrete-continuous structure, where images are distributed along the discrete variables.

5. Budget

In this chapter the cost of the project will be analyzed. We will take into account the human resources, equipment and indirect costs.

- **Human resources:** The human resources of the project is only constituted by a telecommunications engineering student, which we assume to have a salary of 10€/h. A 18 ECTS thesis, corresponds to 450 hours of work, so the cost of human resources are **4500€**.
- **Equipment:** A computer has been used to work on the thesis. If we assume that it costs 850€ and has a 5 years lifespan, using a 10% depreciation, the estimated depreciation for the project duration (4 months) are **51€**.
- **Indirect costs:** For the indirect costs we will take into account the cost of electricity and the access to the Internet. We assume that these costs are 15€ and 20€ per month, respectively. The duration of the project are 4 months so the estimated indirect costs are **140€**.

Concept	**Cost**
Human resources	4500€
Equipment	51€
Indirect costs	140€
Total cost	**4691€**

Table 5.1: Estimated total cost of the project.

6. Conclusion and future development

After the completion of this project, some conclusions can be extracted regarding the research questions of the thesis. The main conclusion of the project is related to the first research question: *Can a GAN be used to sort the samples appropriately without having access to the class annotations?*. Obtained results have shown that it is possible to cluster images using GANs, but only according to low-level features (according to the experiments). To be able to cluster according to high-level features, making low-level features be the same in all images could work, for example extracting the background from CIFAR-10 images.

In the case of clustering of images by their dominant color, results obtained are promising since nearby colors in RGB or CIELAB color space are clustered together and some colors are mainly assigned to a single cluster.

Concerning the second research question: *Once trained, can the GAN be used to label a query image by searching for the closest match in the latent space?*, a trained model can be used to label a query image by using it as $\mathcal{E}$'s input, getting the assigned cluster and mapping it to its corresponding label or, if ground-truth labels for the training set are available, applying KNN. Of course, if the clustering quality is poor, the annotation of query images will be weak.

6.1 Future work

One option to enable the clustering of more complex datasets could be to change the input latent space or to add a term to the loss function (see eq. 2.6) that enforces G to differentiate between high-level features. For the task of colors clustering using GANs, future work could be to use a pixel-based approach instead of using patches, thoroughly collecting a bigger dataset or modifying networks architecture, hyperparameters and number of clusters.

Bibliography

[1] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: *Advances in neural information processing systems* (2014), pp. 2672–2680.

[2] Xi Chen et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *Advances in Neural Information Processing Systems* (2016), pp. 2172–2180.

[3] S. Mukherjee et al. "ClusterGAN : Latent space clustering in generative adversarial networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (2019), pp. 4610–4617.

[4] Stuart P. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[5] Anant Ram et al. "A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases". In: *International Journal of Computer Applications* 3 (June 2010). DOI: 10.5120/739-1038.

[6] Douglas Reynolds. "Gaussian Mixture Models". In: *Encyclopedia of Biometrics* (Jan. 2008). DOI: 10.1007/978-0-387-73003-5_196.

[7] Olcay Akman et al. "Chapter 11 - Data Clustering and Self-Organizing Maps in Biology". In: *Algebraic and Combinatorial Computational Biology*. Ed. by Raina Robeva and Matthew Macauley. MSE/Mathematics in Science and Engineering. Academic Press, 2019, pp. 351–374. ISBN: 978-0-12-814066-6. DOI: https://doi.org/10.1016/B978-0-12-814066-6.00011-8.

[8] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2015. arXiv: 1511.06434 [cs.LG].

[9] Junyuan Xie, Ross Girshick, and Ali Farhadi. "Unsupervised deep embedding for clustering analysis". In: *International conference on machine learning* (2016), pp. 478–487.

[10] Bo Yang et al. "Towards k-means-friendly spaces: Simultaneous deep learning and clustering". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (2017), pp. 3861–3870.

[11] Jianwei Yang, Devi Parikh, and Dhruv Batra. "Joint unsupervised learning of deep representations and image clusters". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 5147–5156.

[12] Nairouz Mrabah et al. “Deep Clustering with a Dynamic Autoencoder: From Reconstruction towards Centroids Construction”. In: *arXiv preprint arXiv:1901.07752* (2019).

[13] Diederik Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: Dec. 2014.

[14] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[15] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein GAN”. In: (Jan. 2017).

[16] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[17] Joost van de Weijer, Cordelia Schmid, and Jakob Verbeek. “Learning Color Names from Real-World Images”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), pp. 1–8.

[18] Thomas Hofmann. “Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization”. In: *Advances in Neural Information Processing Systems* (2000), pp. 914–920.

[19] *ISO 11664-4:2008(E): Colorimetry — Part 4: CIE 1976 L*a*b* Colour space*. Standard. 2007.

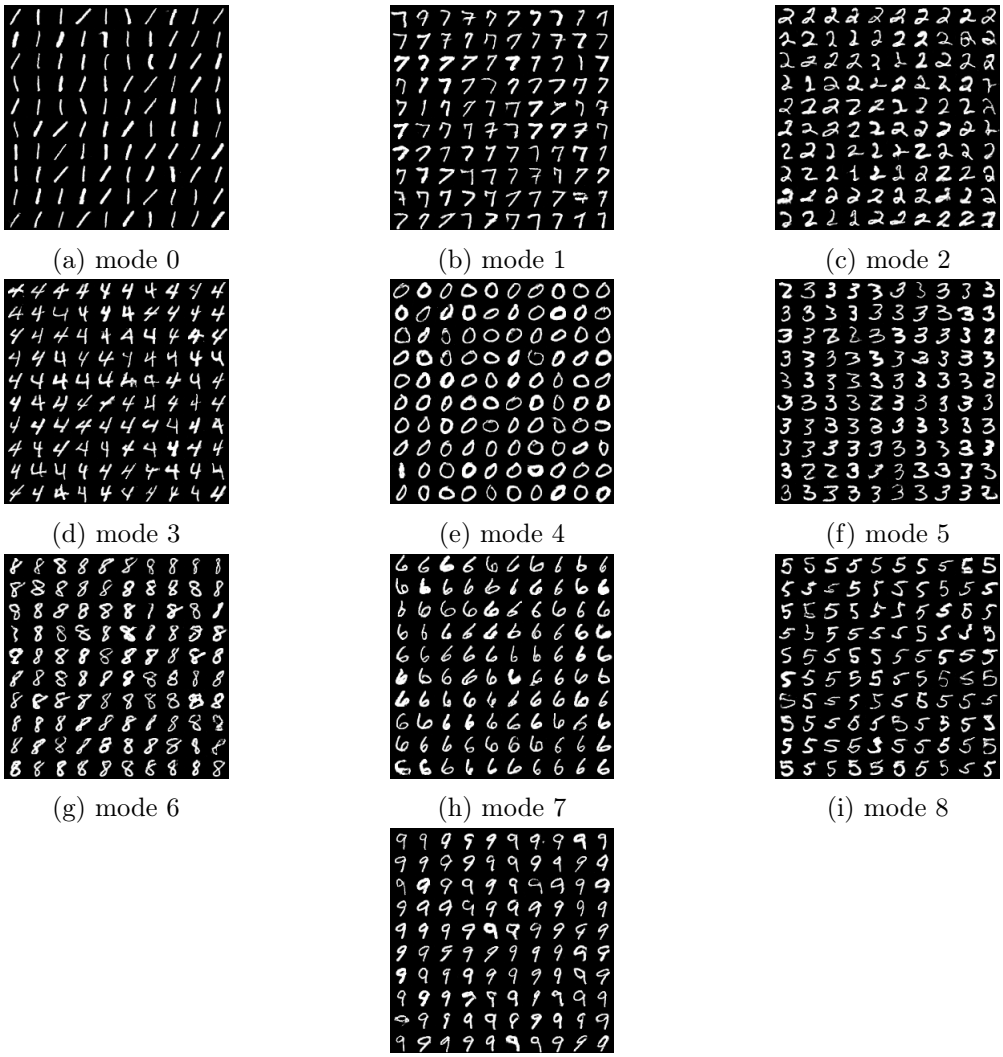
[20] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[21] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 2014).

[22] Marwan Hassani and Thomas Seidl. “Using internal evaluation measures to validate the quality of diverse stream clustering algorithms”. In: *Vietnam Journal of Computer Science* 4 (2017), pp. 171–183. DOI: https://doi.org/10.1007/s40595-016-0086-9.

[23] Thomas M. Cover and Peter E. Hart. “Nearest neighbor pattern classification”. In: *IEEE Trans. Inf. Theory* 13 (1967), pp. 21–27.

Appendix



(a) mode 0 (b) mode 1 (c) mode 2

(d) mode 3 (e) mode 4 (f) mode 5

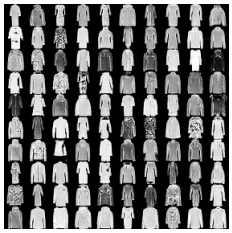
(g) mode 6 (h) mode 7 (i) mode 8

(j) mode 9

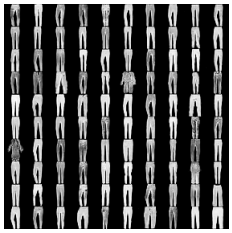
Figure 6.1: Generated MNIST digits from distinct modes

(a) mode 0

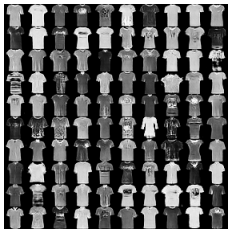
(b) mode 1



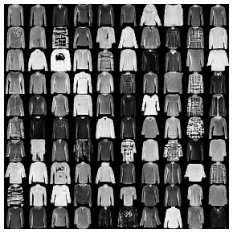
(c) mode 2



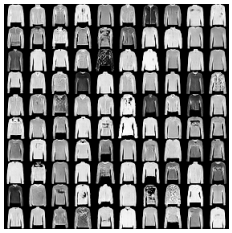
(d) mode 3



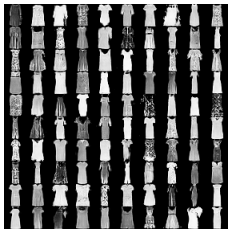
(e) mode 4



(f) mode 5



(g) mode 6

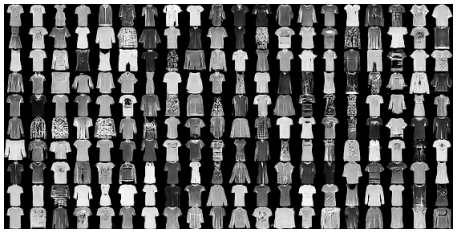


(h) mode 7

(i) mode 8

(j) mode 9

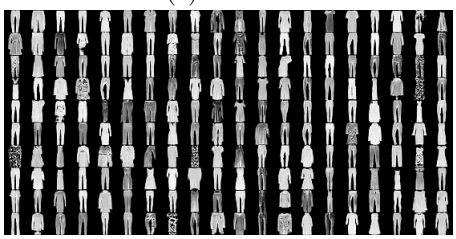
Figure 6.2: Generated Fashion-MNIST items from distinct modes



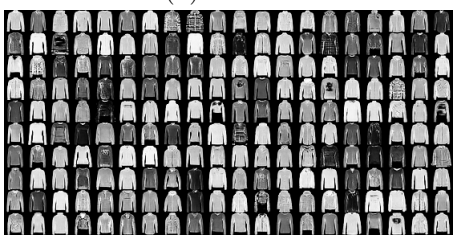
(a) mode 0

(b) mode 1

(c) mode 2



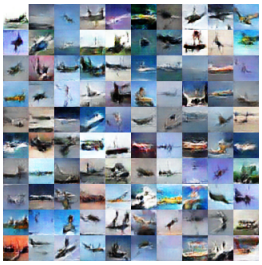
(d) mode 3



(e) mode 4

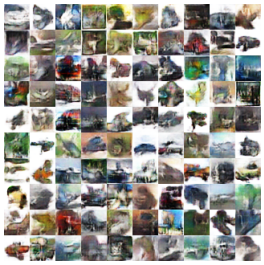
Figure 6.3: Generated Fashion-5 items from distinct modes

(a) mode 0



(b) mode 1

(c) mode 2

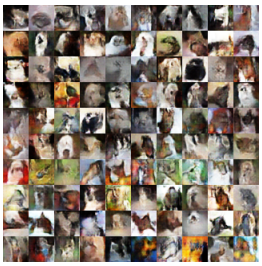


(d) mode 3

(e) mode 4

(f) mode 5

(g) mode 6

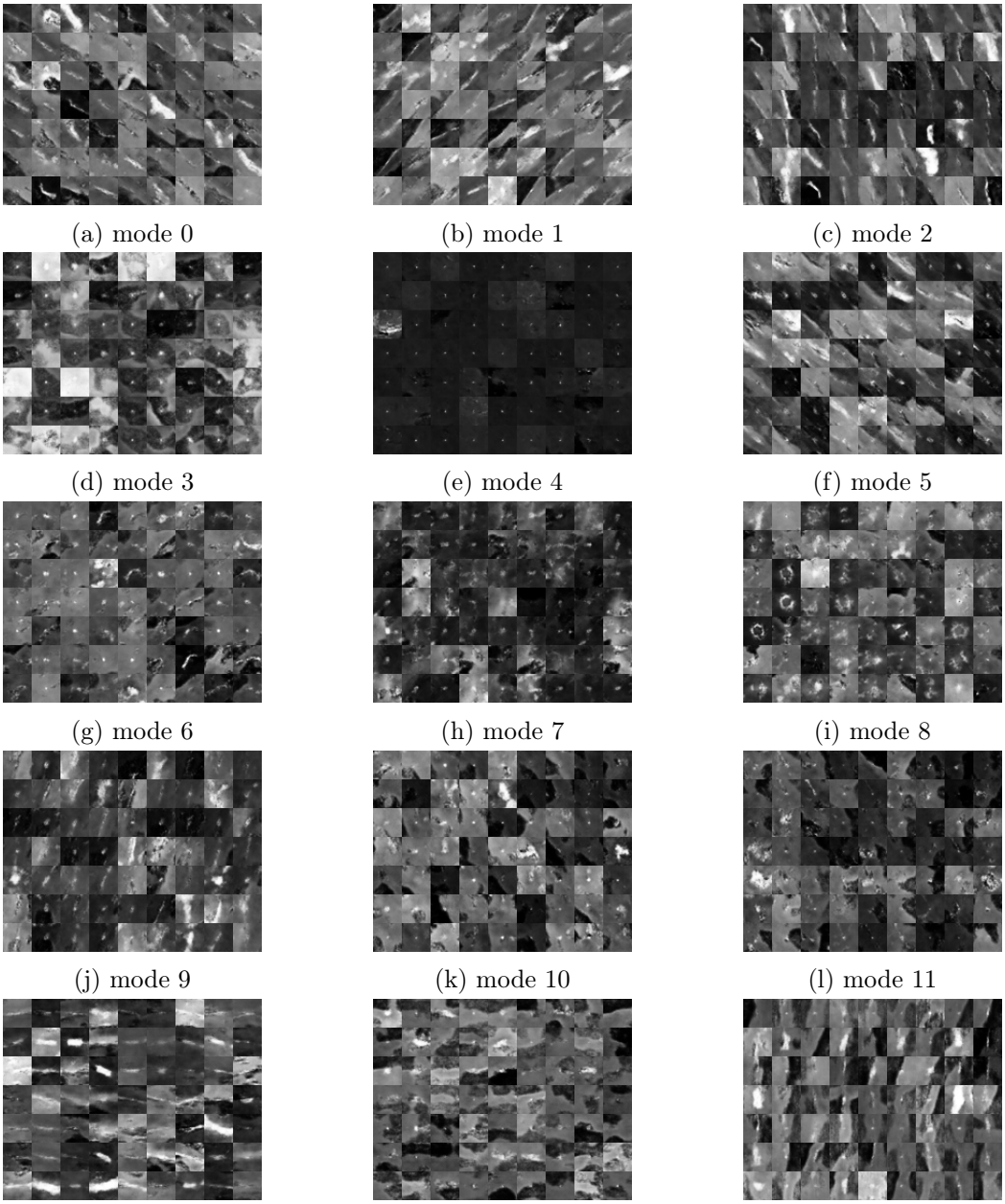


(h) mode 7

(i) mode 8

(j) mode 9

Figure 6.4: Generated CIFAR-10 categories from distinct modes



(a) mode 0 (b) mode 1 (c) mode 2

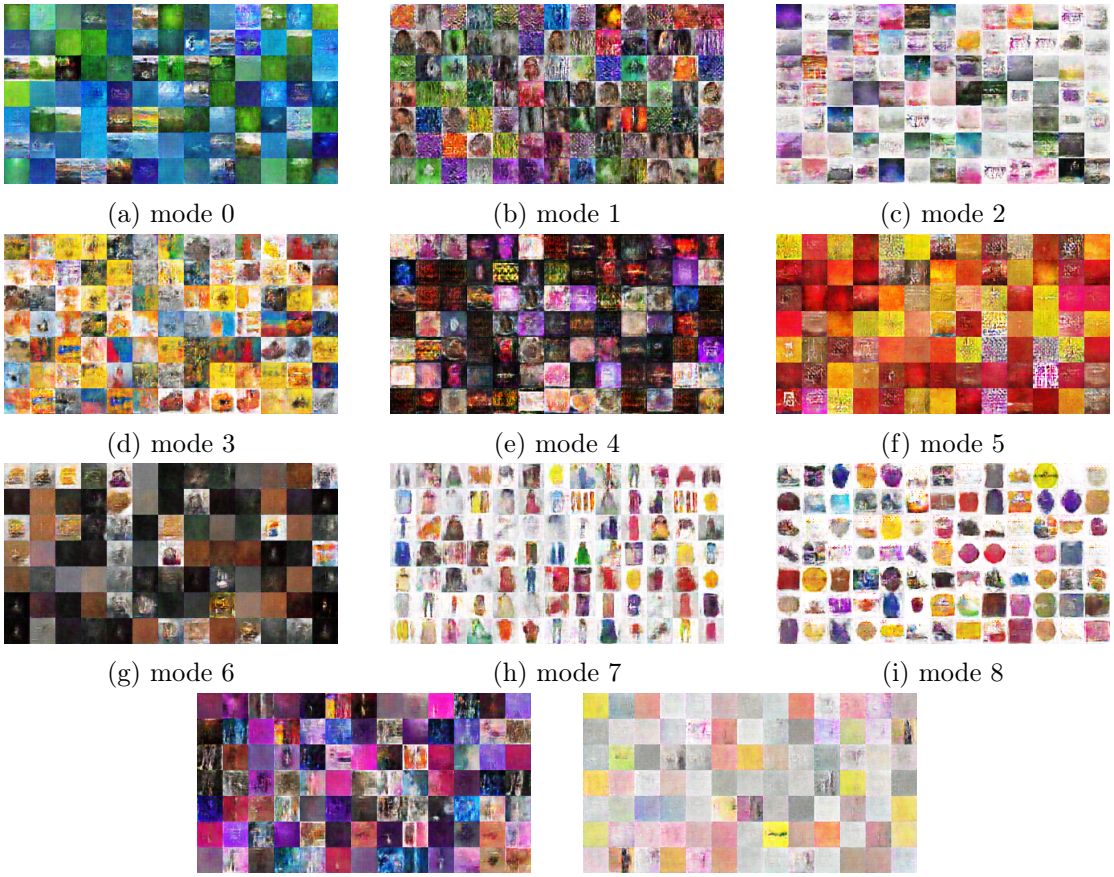
(d) mode 3 (e) mode 4 (f) mode 5

(g) mode 6 (h) mode 7 (i) mode 8

(j) mode 9 (k) mode 10 (l) mode 11

(m) mode 12 (n) mode 13 (o) mode 14

Figure 6.5: Generated images from distinct modes of the Termisk dataset



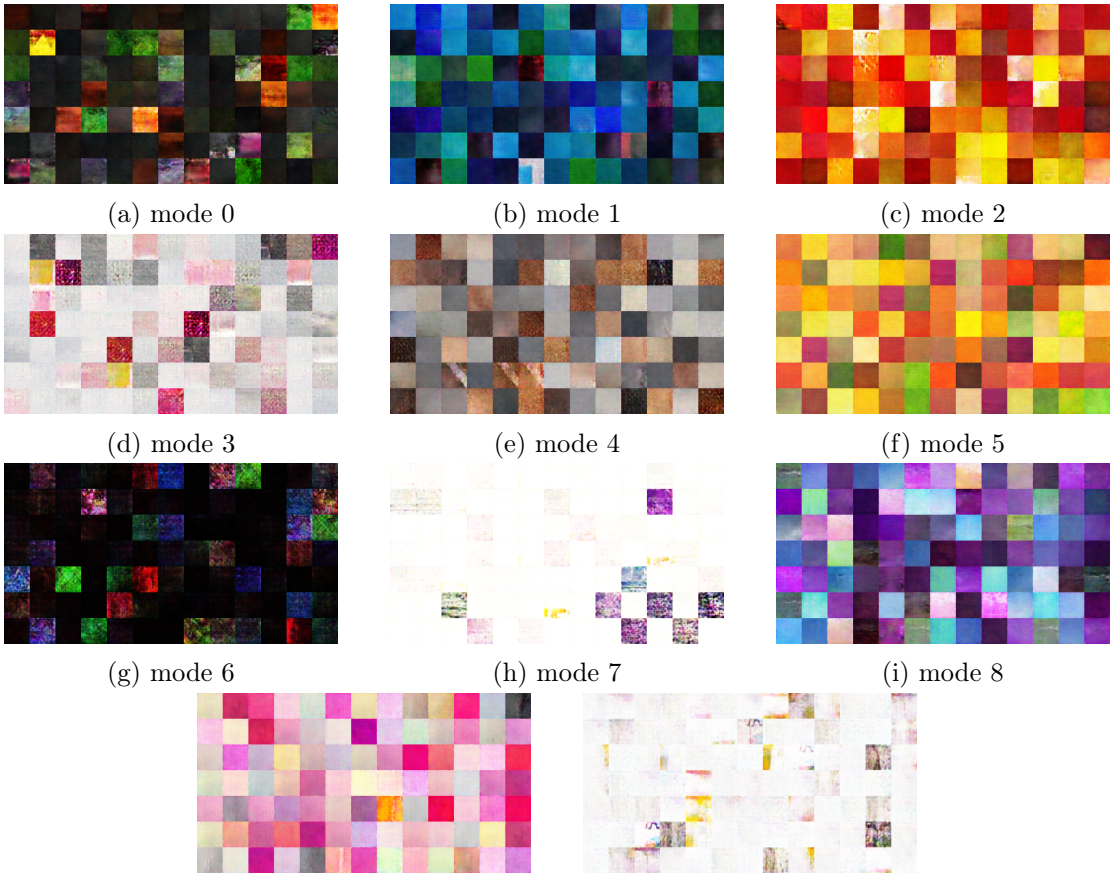
(a) mode 0 (b) mode 1 (c) mode 2

(d) mode 3 (e) mode 4 (f) mode 5

(g) mode 6 (h) mode 7 (i) mode 8

(j) mode 9 (k) mode 10

Figure 6.6: Generated images from distinct modes using RGB resized images



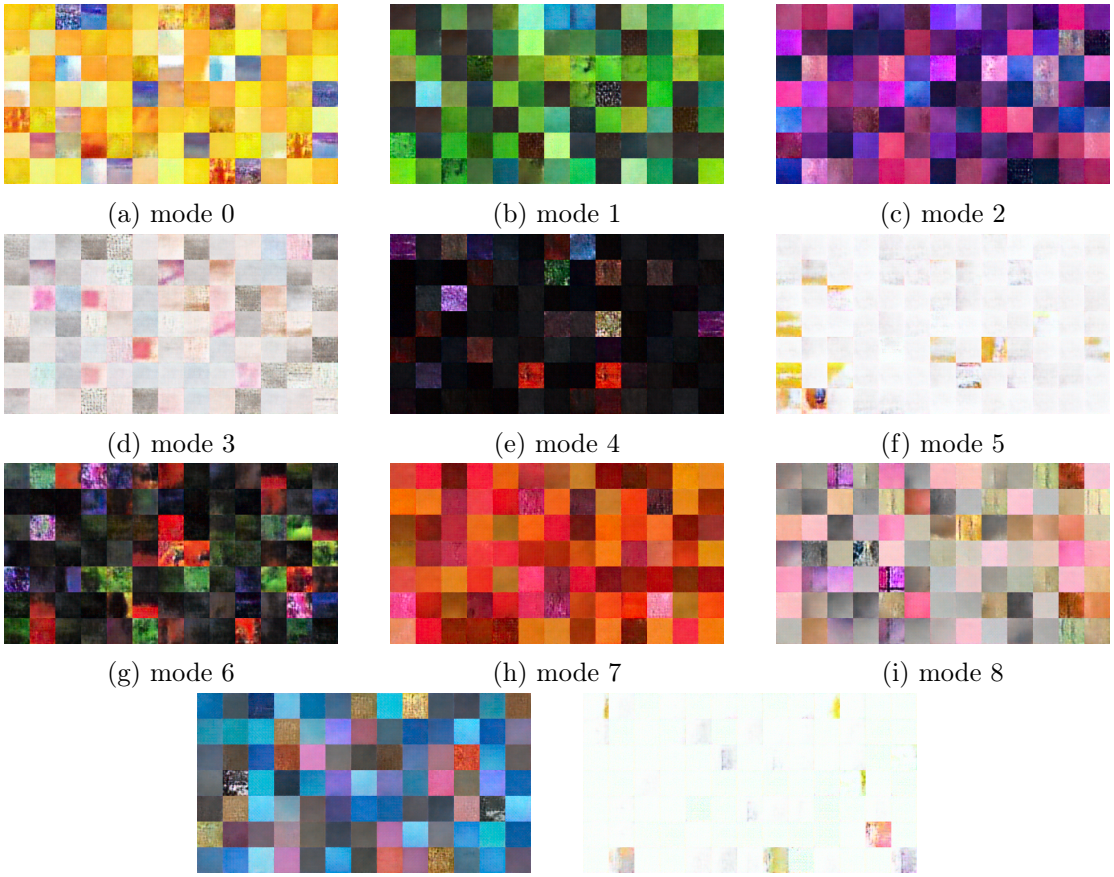
(a) mode 0 (b) mode 1 (c) mode 2

(d) mode 3 (e) mode 4 (f) mode 5

(g) mode 6 (h) mode 7 (i) mode 8

(j) mode 9 (k) mode 10

Figure 6.7: Generated images from distinct modes using RGB patches and 11 clusters



(a) mode 0 (b) mode 1 (c) mode 2

(d) mode 3 (e) mode 4 (f) mode 5

(g) mode 6 (h) mode 7 (i) mode 8

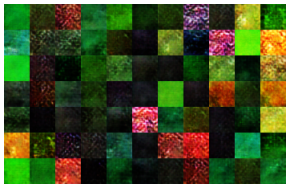
(j) mode 9 (k) mode 10

Figure 6.8: Generated images from distinct modes using CIELAB patches and 11 clusters

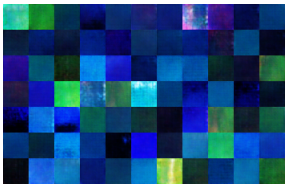
(a) mode 0

(b) mode 1

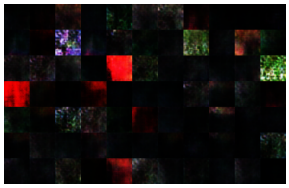
(c) mode 2



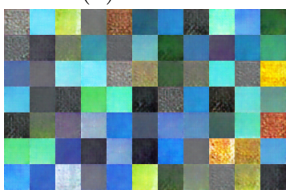
(d) mode 3



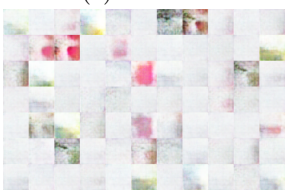
(e) mode 4



(f) mode 5

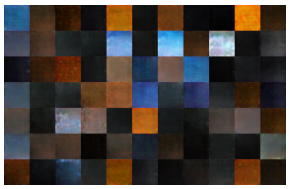


(g) mode 6

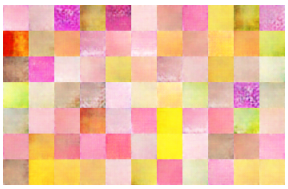


(h) mode 7

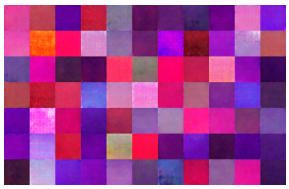
(i) mode 8



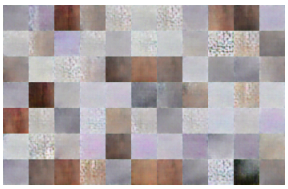
(j) mode 9



(k) mode 10



(l) mode 11



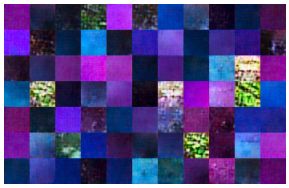
(m) mode 12

Figure 6.9: Generated images from distinct modes using RGB patches and 13 clusters

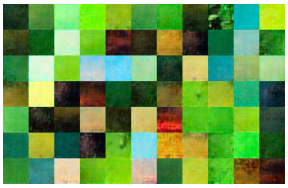
(a) mode 0

(b) mode 1

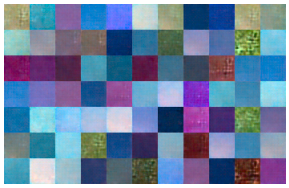
(c) mode 2



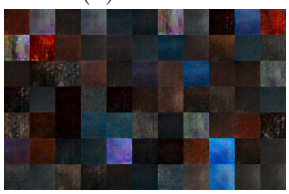
(d) mode 3



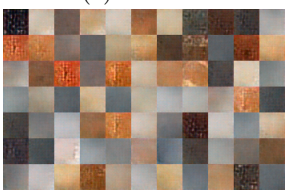
(e) mode 4



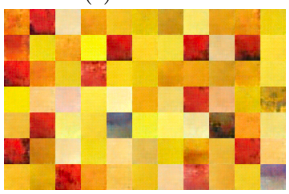
(f) mode 5



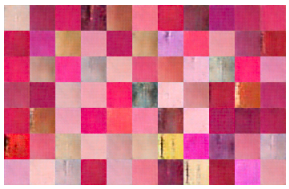
(g) mode 6



(h) mode 7

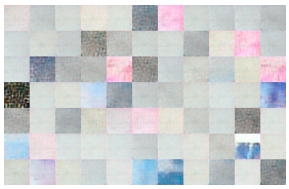


(i) mode 8



(j) mode 9

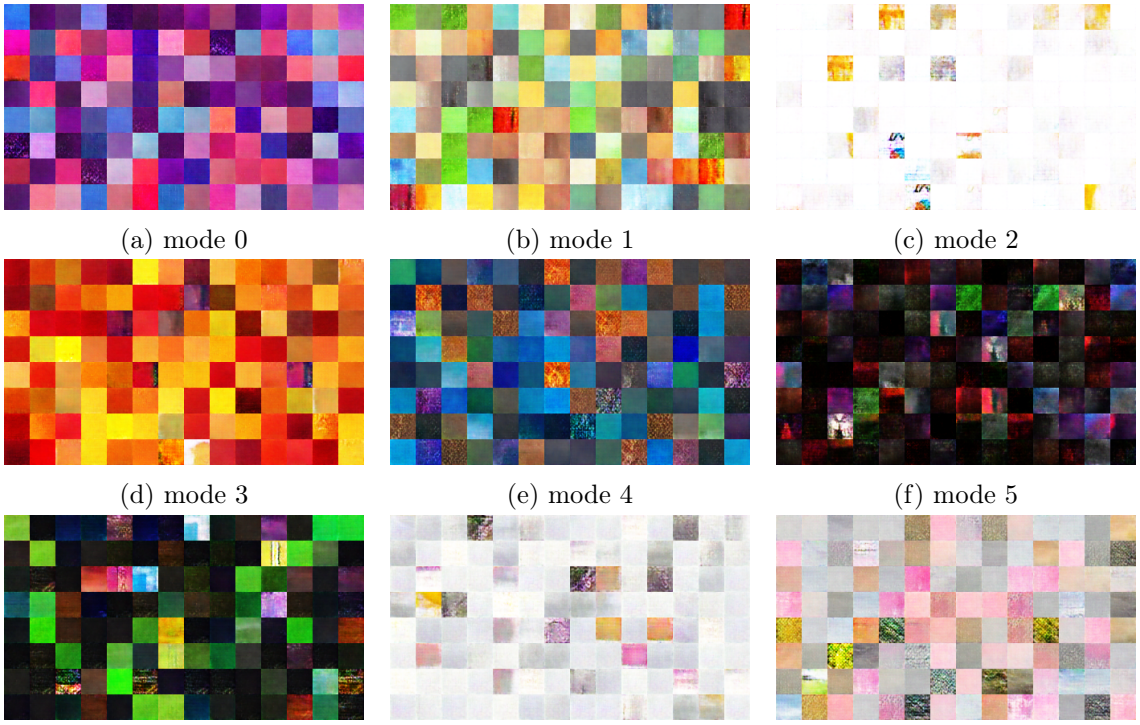
(k) mode 10



(l) mode 11

(m) mode 12

Figure 6.10: Generated images from distinct modes using CIELAB patches and 13 clusters

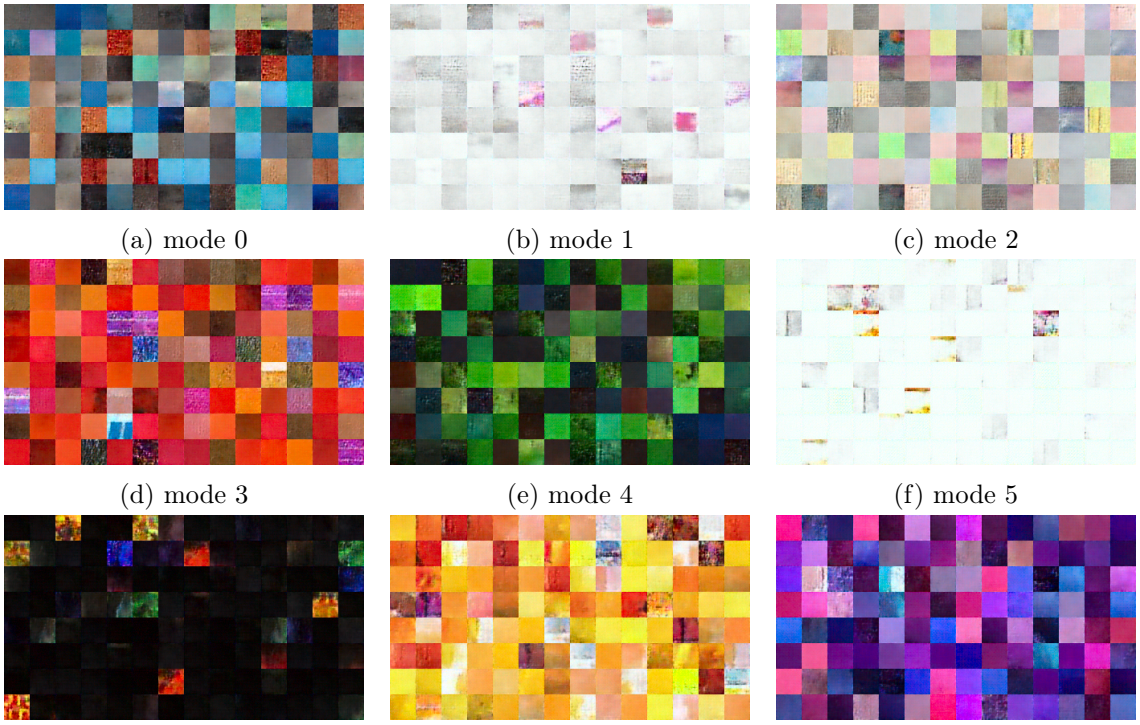


(a) mode 0 (b) mode 1 (c) mode 2

(d) mode 3 (e) mode 4 (f) mode 5

(g) mode 6 (h) mode 7 (i) mode 8

Figure 6.11: Generated images from distinct modes using RGB patches and 9 clusters



(a) mode 0 (b) mode 1 (c) mode 2

(d) mode 3 (e) mode 4 (f) mode 5

(g) mode 6 (h) mode 7 (i) mode 8

Figure 6.12: Generated images from distinct modes using CIELAB patches and 9 clusters